REMARKS/ARGUMENTS

Amendments were made to the specification to correct errors and to clarify the specification. No new matter has been added by any of the amendments to the specification.

Claims 1-24 are pending in the present application. Claims 1, 2, 4, 7, 9, 10, 12, 15, 17, 18, 20 and 23 have been amended herewith. Reconsideration of the claims is respectfully requested.

I. Objection to Specification

The Examiner objected to the Specification, stating:

(i) certain acronyms used in page 8 are not defined; and

(ii) the 'hardware console' described on page 12 has an incorrect reference numeral.

Applicants have amended the Specification at page 8 to include the well-known acronym definitions. Applicants have amended the Specification at page 12 to correct the reference numeral.

Therefore, the objection to the Specification has been overcome.

II. 35 U.S.C. § 112, First Paragraph

The Examiner rejected Claims 4, 12 and 20 under 35 U.S.C. § 112, first paragraph, in that the claims are alleged to contain subject matter which was not described in the specification, as the "*location* at which the context of the stopped processor was not described within the specification". Applicants respectfully submit that Claims 4, 12, and 20 do not recite or otherwise contain any type of 'location' feature, and instead recite "wherein context for the processor in a slave loop is stored to form a stored context". As can be seen, Claim 4 is directed to storing context for 'the processor in the slave loop', and this is described at Specification page 15, lines 9-30 and depicted in Figure 3, block 314.

Therefore, the rejection of Claims 4, 12 and 20 under 35 U.S.C. § 112, first paragraph has been overcome.

III. 35 U.S.C. § 101

The Examiner rejected Claims 17 and 23 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. This rejection is respectfully traversed.

The Examiner states that Claims 17 and 23 impermissibly encompass transmission-type media. While the Examiner provides no statutory authority for the premise that transmission-type media are non-statutory, and Applicants deny such assertion, Applicants have in any event amended Claims 17 and 23 such that this case can expeditiously pass to issuance.

Therefore, the rejection of Claims 17 and 23 under 35 U.S.C. § 101 has been overcome.

IV. 35 U.S.C. § 102, Anticipation

The Examiner rejected Claims 1, 6, 9, 14, 17 and 22 under 35 U.S.C. § 102(b) as being anticipated by Intel *("Itanium™ Processor Floating-point Software Assistance and Floating-point Exception Handling." January 2000)*. This rejection is respectfully traversed.

For a prior art reference to anticipate in terms of 35 U.S.C. 102, every element of the claimed invention must be identically shown in a single reference. *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). Applicants will now show that every element of the claimed inventions recited in Claims 1, 6, 9, 14, 17 and 22 is not identically shown in the cited Intel reference, and therefore these claims are not anticipated by the cited reference.

Claim 1 recites 4 steps:

(i) identifying an exception vector to form an identified exception vector when control is passed from an operating system to the firmware;
(ii) saving the identified exception vector to form a saved exception vector;
(iii) replacing the identified exception vector with a substitute vector; and
(iv) restoring the saved exception vector when control is returned to the operating system to form a restored exception vector, wherein the restored exception vector continues execution.

In rejecting step (i) of Claim 1, the Examiner merely alleges that the cited reference teaches:

"means and instructions for identifying an exception"

Notably absent is any allegation or showing that the cited reference teaches that such identifying step is done 'when control is passed from an operating system to the firmware' (emphasis added by Applicants), as expressly recited in step (i) of Claim 1. The cited passage (Intel page 2-2, paragraphs 4-7) that is used in rejecting step (i) of Claim 1 describes an emulation library and associated processing by such library. As can be seen by Intel page 1-3, this emulation library is a part of the operating system kernel, and this operating system kernel is not described as being firmware. In addition, as was commonly known to those of ordinary skill in the art – as evidenced by the kernel definition provided in Attachment I attached hereto – an operating system kernel is traditionally recognized to be software (i.e. not firmware). This is particularly evident on page 2 of Attachment I, where in the figure on the right side of the page the kernel and the firmware are shown to be separate elements. It is therefore urged that Intel's discussion of an operating system kernel functionality does not describe any type of operation that occurs 'when control is passed from an operating system to the firmware', as expressly recited in Claim 1.

Further, the cited Intel reference itself describes firmware to be a separate component from this emulation library (Intel pages 7-1 thru 7-2). While this library may be initially loaded from firmware (Intel page 7-1 thru 7-2), this is an initial part of initializing the library to make it available for use as a part of the operating system, and it is configured as a traditional EFI driver for subsequent access by the operating system using traditional driver calls (page 7-1, last paragraph). Restated, when being used in operation, this emulation library and the functions performed thereby do not occur 'when control is passed from an operating system to the firmware', as expressly recited in Claim 1. Therefore, the cited passage at Intel page 2-2, paragraphs 4-7 that describes the detailed operation of this emulation library do not occur when control is passed from an operating system to the firmware. Accordingly, as every element of the claimed invention recited in Claim 1 is not identically shown in a single reference – and in particular every element of step (i) of Claim 1 - it is urged that Claim 1 is not anticipated by the cited Intel reference.

Still further with respect to Claim 1, and specifically with respect to step (ii) of such claim, the Examiner states in rejecting this aspect of Claim 1 that Intel teaches:

> "means and instructions for saving the identified exception"

As can be seen, the Examiner merely alleges that a single identified exception is saved. In contrast, step (ii) of Claim 1 is not merely directed to saving a single exception, but instead is directed to saving an exception *vector*. The Examiner has not alleged, nor does the cited reference teach, the saving of an exception vector. In any event, Claim 1 has been amended per the Specification description at page 14, lines 13-14 in order to further differentiate the claimed exception vector processing from the teachings of the cited Intel reference. In particular, Claim 1 now recites that the exception vector – which is identified, saved, replaced and restored - contains multiple objects for processing both traps and interrupts for the operating system, and thus this exception vector is very different from a single exception. Thus, it is further urged that Claim 1 is not anticipated by the cited reference.

Still further with respect to Claim 1, and specifically with respect to step (iii) of such claim, the Examiner states in rejecting this aspect of Claim 1 that Intel teaches:

> "means and instructions replacing the exception vector with a *substitute exception* (page 2-2, ¶5 and ¶6)(The emulation library can call the kernel exception handler or a user level floating point exception handler to continue processing the exception" (emphasis added by Applicants to emphasis that this is with respect to a single exception)

Again, this description describes processing of a *single exception* by either of two handlers – a kernel exception handler or a user level exception handler. This passage does not describe any type of replacing of an *exception vector* with another *exception vector*. Thus, Claim 1 is further shown to have been erroneously rejected as there are additional claimed features which are not identically shown in a single reference.

Still further with respect to Claim 1, and specifically with respect to step (iv) of such claim, the Examiner states in rejecting this aspect of Claim 1 that Intel teaches:

> "means and instructions for restoring the saved exception when control when control is returned to the operating system (page 2-2, ¶4-7)(Control is returned to the operating system kernel exception handler before execution of the running program is resumed)".

As can be seen, the Examiner characterizes this cited passage as teaching a return of control to the operating system kernel exception handler. This passage does not describe a specific step of *restoring a saved exception vector* when this control is returned, as expressly recited in Claim 1. Thus, Claim 1 is further shown to have been erroneously rejected as there are additional claimed features which are not identically shown in a single reference.

In summary, the passages cited by the Examiner in rejecting Claim 1 are all directed to the processing of a single exception by an operating system kernel, and such passages (1) do not describe any operations that occur *when control is passed from an operating system to the firmware*, and (2) do not describe any saving, replacing or restoring of an *exception vector*. Accordingly, it is urged that Claim 1 is not anticipated by the cited reference for the numerous reasons identified above.

Applicants traverse the rejection of Claim 6 for reasons given above with respect to Claim 1 (of which Claim 6 depends upon).

With respect to Claims 9 and 17, Applicants traverse for similar reasons to those given above with respect to Claim 1.

With respect to Claims 14 and 22, Applicants traverse for similar reasons to those given above with respect to Claim 6.

Therefore, the rejection of Claims 1, 6, 9, 14, 17 and 22 under 35 U.S.C. § 102(b) has been overcome.

V. 35 U.S.C. § 103, Obviousness

The Examiner rejected Claims 2, 4, 5, 7, 8, 10, 12, 13, 15, 16, 18, 20, 21, 23 and 24 under 35 U.S.C. § 103 as being unpatentable over Intel *("Itanium™ Processor Floating-point Software Assistance and Floating-point Exception Handling." January 2000)* in view of Jones *(Jones, Steve. "Using spinlocks in a symmetric multiprocessing environment." Tech Specialist, v2, n10, pg 15(6), Oct 1991).* This rejection is respectfully traversed.

Applicants initially traverse the rejection of Claims 2, 4 and 5 for reasons given above with respect to Claim 1 (of which Claims 2, 4 and 5 depend upon), and urge that the newly cited Jones reference does not overcome the teaching deficiencies identified above with respect to Claim 1.

Further with respect to Claim 2 (and dependent Claims 4 and 5), such claim recites "responsive to a processor, other than an associated processor associated with the save interrupt vector, generating an error for the identified exception vector replaced by the substitute vector, placing the processor in a slave loop until the saved exception vector is restored". As can be seen, Claim 2 is directed to steps that occur in response to an error generated by another processor other than the processor associated with the save interrupt vector, including a step of placing this another processor in a slave loop *until the saved exception vector is restored*". In rejecting Claim 2, the Examiner cites Jones spinlock as being equivalent to the claimed slave loop, and states that Jones teaches use of a spinlock to pause the execution of a BIOS routine by processes other than the one currently processing the BIOS routine. Importantly, the Examiner does not allege – and the cited reference in fact does not teach or suggest – the specific claimed feature of (1) placing the processor in a slave loop *until the saved exception vector is restored*, or (2) that such slave loop placement is done in response a processor *generating an error for the exception vector*. Rather, the Examiner merely alleges, and the reference only teaches, a generic use of a spin lock to guard a data structure. Thus, the Examiner has failed to establish a prima facie showing of obviousness[1]. Accordingly, the burden has not shifted to Applicants to rebut this obviousness assertion[2]. In addition, as a proper prima facie showing of obviousness has not been established, Claim 2 has been erroneously rejected under 35 USC 103[3].

---

[1] To establish prima facie obviousness of a claimed invention, all of the claim limitations must be taught or suggested by the prior art. MPEP 2143.03. *See also, In re Royka*, 490 F.2d 580 (C.C.P.A. 1974).

[2] In rejecting claims under 35 U.S.C. Section 103, the examiner bears the initial burden of presenting a prima facie case of obviousness. *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992). Only if that burden is met, does the burden of coming forward with evidence or argument shift to the applicant. *Id.*

[3] If the examiner fails to establish a prima facie case, the rejection is improper and will be overturned. *In re Fine*, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988).

Further with respect to Claim 4 (and dependent Claim 5), it is urged that none of the cited references teach or suggest the claimed feature of "wherein context for the processor in the slave loop is stored to form a stored context". In rejecting this aspect of Claim 4, the Examiner states:

> "The use of a spinlock to pause the execution of a *processor inherently requires that the state of the processor be saved* because, while the processor keeps checking the state of the lock, it must use registers (memory locations local to the processor that are used to perform operations) previously occupied by the previous task being executed. The contents of the registers that correspond to the information relating to the exception must be stored in RAM or some other storage medium to allow the processor to continue with that exception, once the lock is held by that processor. Once that lock is held, the *processor must inherently load the saved context back* into the registers in order to continue its previous operation." (emphasis added by Applicants)

Applicants urge error in such inherency assertion. "Inherency . . . may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient." *In re Oelrich*, 666 F.2d 578, 581, 212 USPQ 323, 326 (CCPA 1981) (quoting *Hansgirg v. Kemmer*, 102 F.2d 212, 214, 40 USPQ 665, 667 (CCPA 1939)). "To establish inherency," the Federal Circuit recently stated, "the extrinsic evidence `must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill.'" *In re Robertson*, 169 F.3d 743, 745 [49 USPQ2d 1949] (Fed. Cir. 1999); see also *Continental Can Co. U.S.A., Inc. v. Monsanto Co.*, 948 F.2d 1264, 1268 [20 USPQ2d 1746] (Fed. Cir. 1991). Such inherency may not be established by "`probabilities or possibilities.'" *Continental Can*, 948 F.2d at 1269 (quoting In re Oelrich, 666 F.2d 578, 581 [212 USPQ 323] (C.C.P.A. 1981)). As evidenced by Attachment II attached hereto, a spinlock is a lock where the thread simply waits in a loop ("spins") repeatedly checking until the lock becomes available. As the thread remains active but isn't performing a useful task, the use of such a lock is a kind of busy waiting. Spinlocks are efficient if threads are only likely to be blocked for a short period of time, as they *avoid overhead from operating system process re-scheduling or context switching*. For this reason, spinlocks are often used inside operating system kernels. Thus, it is shown that context switching by a processor is not necessarily present when using a spinlock, as alleged by the Examiner, and in fact such context switching is not done when using a spinlock. Therefore, the missing claimed features identified above as not be expressly taught or suggested by the cited references are also not inherent. Thus, Claim 4 (and dependent Claim 5) is further

shown to have been erroneously rejected as there are additional claimed features not taught or suggested by the cited references.

Applicants further traverse the rejection of Claims 10, 12, 13, 18, 20 and 21 for similar reasons to those given above with respect to Claims 2, 4 and 5.

With respect to Claim 7 (and dependent Claim 8), such claim recites "restoring the exception vector when control is returned to the operating system, wherein processors, other than a particular processor creating the exception vector, encountering the substitute code are suspended until control is returned to the operating system". As can be seen, per the features of Claim 7, (1) the exception vector is restored *when control is returned to the operating system*, and (2) processors encountering substituted code are suspended. In rejecting this aspect of Claim 7, the Examiner cites Intel as teaching that control is returned, but fails to allege any teaching that an exception vector is restored when such control is returned, as expressly recited in Claim 7 (as further described above regarding Intel's teaching deficiencies with respect to step (iv) of Claim 1). Still further, the Examiner cited Jones' use of a spinlock to pause execution of a BIOS routine by other processes, but fails to allege any teaching that processors are suspended *when encountering substituted code*. This is likely because Jones' teachings with respect to spinlocks are not in any way directed to *substituted* code, but instead are used to protect preexisting data structures (Jones page 2, lines 5-6 et seq.). Accordingly, for at least these two reasons alone, the Examiner has failed to properly establish a prima facie case of obviousness and therefore the burden has not shifted to Applicants to rebut such improper obviousness assertion. In addition, as a proper prima facie showing of obviousness has not been established, Claim 7 (and dependent Claim 8) has been erroneously rejected under 35 USC 103, *In re Fine, Id.*

Applicants traverse the rejection of Claims 15, 16, 23 and 24 for similar reasons to those given above with respect to Claims 7 and 8.

Therefore, the rejection of Claims 2, 4, 5, 7, 8, 10, 12, 13, 15, 16, 18, 20, 21, 23 and 24 under 35 U.S.C. § 103 has been overcome.

VI. 35 U.S.C. § 103, Obviousness

The Examiner rejected Claims 3, 11 and 19 under 35 U.S.C. § 103 as being unpatentable over Intel *("Itanium™ Processor Floating-point Software Assistance and Floating-point Exception Handling." January 2000)* in view of Jones *(Jones, Steve. "Using spinlocks in a symmetric multiprocessing environment." Tech Specialist, v2, n10, pg 15(6), Oct 1991)* and in further view of IBM *(IBM Technical Bulletin: NNRD447149. "Method to Prevent Multiple Processes After Taking Exceptions to Enter Open Firmware in a Symmetrical Multiprocessor Machine", Published 01 July 2001).* This rejection is respectfully traversed.

With respect to Claim 3 (and similarly for Claims 11 and 19), Applicants traverse such rejection for similar reasons to those given above with respect to Claim 2 (of which Claim 3 depends upon), and urge that the newly cited IBM reference does not overcome the teaching/suggestion deficiencies identified above with respect to Claim 2.

Therefore, the rejection of Claims 3, 11 and 19 under 35 U.S.C. § 103 has been overcome.

VII. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: November 15, 2006

Respectfully submitted,

/Wayne P. Bailey/

Wayne P. Bailey
Reg. No. 34,289
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants